Estimate the Total Completion Time of the Workload

Muhammad Amjad*[1], Waqas Ahmad[2], Zia Ur Rehman[3], Waqar Hussain[4], Syed Badar Ud Duja[5], Bilal Ahmed[6], Usman Ali[7], M. Abdul Qadoos[8], Ammad Khan[9], M. Umar Farooq Alvi[10]
College of Information and Computer Taiyuan University of Technology
Taiyuan, Shanxi, China

***Abstract*—The business intelligence workload is required to serve analytical process. The data warehouses have a very large collection of digital data. The large collection of digital data is required to analytical process within the perplexing workload. The main problem for perplexing workload is to estimate the total completion time. Estimate total completion time is required when workload is executed as a batch of queries. To estimate the queries according to their interaction aware scheme because queries are run in batches. The database administrators often require to perceive how much longer time for business intelligence workloads will take to complete. This question ascends, when database administrator entails to accomplish workloads within existing time frame. The database system executes mixes of multiple queries concurrently. We would rather measure query interactions of a mix than practiced approach to consider each query separately. A novel approach as a estimate framework is presented to estimate running time of a workload based on experiment driven modeling coupled with workload simulation. An estimation framework is developed which has two major parts offline phase and online phase. Offline phase collects the experiments sampling of mixes which has different query types. To find the good accuracy for estimating the running time of the workload by evaluation with TPC-H queries on PostgreSQL.**

***Keywords*—Query interactions; estimate time; running time**

I. Introduction

OLAP workload has long-running queries that has to execute database system at different time period repeatedly. These batches of queries execution time range sometimes from minuts to hours. The database administrator wants to accomplish business intelligent workloads within time frame but due to indeterminacies of queries execution time can't fulfill her requirement. Resource contentions are a reason which effects response time of a query. Therefore, to measure query interactions in a mix is essential, a phenomenon that query execution might be hastened or hindered by parallel queries [23]. For example the performance of Q_{18} and Q_5 describe in the three mixes m_2, m_3 and m_4. The m_2 presents the positive interaction for Q_{18}. Q_{18} has the average response time 609 seconds while run alone in the system is 624 seconds. On the others hand Q_{18} suffers in mix m_3 due to negative interaction. Q_{18} has the average response time 707 seconds in mix m_3. m_4 is also a positive interaction for Q_5. We need to capture these interactions.

If query interaction in a mix is measured then a database administrator can adjust business intelligent workload within time-bound without flawlessly. The state of the art does not provide any method to estimate the total completion time of a workload. To predict query execution time that is not only use for estimate the total completion time, it is also useful for database other management tasks, sizing, progress monitoring, admission control and query scheduling [2-5]. Recently most of the work focuses on estimating the time for independent query [6-9]. whereas a very little work studies to predict the time for simultaneously running queries [10]. The database systems most often allow simultaneously running queries. Therefore, to estimate execution time for concurrent queries are required. To estimate simultaneously running queries are more important than for independent queries.

The approaches are investigated to building estimation model for estimating the response time of a query running concurrently with other queries. Specifically, the model focuses to the following scenario. The database systems constantly run a mix of M queries simultaneously. Whenever a query is finished execution and exits, the database systems arbitrary manner select a query from the queue to form a new mix. The model is required for estimating the response time of a query at any time point of its running.

In this paper experiment driven approach is used to take into account the query interactions. Experiment driven approach is attainment to build performance static model which estimate the query response time. A dynamic model is manipulate such performance static model for estimating the response time of the newly form mixes. A relevant work uses machine learning technique to estimate performance metrics for queries [7], but authors focus at the single query running in the database system and our motive is concurrently running queries.

Our contributions can be concluded as follows:

- A performance static model is proposed to estimate the query response time in a mx.
- A dynamic model is proposed to manipulte the performance static model for estimating newly formed mixes.

To meaure the impact of the query interactions in a mix, queries are used from TPC-H benchmark with a database system extent of 10GB operating on PostgreSQL. Table I shows average running time of the TPC-H queries in the database system. Table II shows the number of each type of queries in a mix.

The rest of paper organized as follow. Section II present related work . A framework is developed in Section III. Section IV presents evaluation of this approach and conclude in Section V.

TABLE I. RUNNING TIME t_j OF SINGLE QUERY TYPES

Query type	Q_1	Q_4	Q_5	Q_6	Q_8	Q_{10}	Q_{12}	Q_{14}	Q_{18}
Runtime tj (sec)	5085	1414	585	578	598	71	866	573	624

TABLE II. THE AVERAGE RUNNING TIME FOR SIMULTANEOUSLY QUERIES

	Mix	M_1	M_2	M_3	M_4
Q_1	A_{ij}	1	0	0	0
	N_{ij}	5193	0	0	0
Q_2	A_{ij}	1	0	0	1
	N_{ij}	1620	0	0	1620
Q_5	A_{ij}	0	0	0	1
	N_{ij}	5193	0	0	443
Q_6	A_{ij}	1	0	0	0
	N_{ij}	602	0	0	0
Q_8	A_{ij}	0	1	0	1
	N_{ij}	5193	707	0	707
Q_{10}	A_{ij}	0	0	1	0
	N_{ij}	5193	0	78	0
Q_{12}	A_{ij}	0	1	0	0
	N_{ij}	5193	1133	0	0
Q_{14}	A_{ij}	0	0	1	0
	N_{ij}	5193	0	607	0
Q_{18}	A_{ij}	0	1	1	0
	N_{ij}	0	609	609	0

II. REALTED WORK

A. Literature Review

The main method of the analytical model is to obtain the query response time by performing a detailed analysis of the query process of the database system. The core research content is (1) the speed at which the system executes the query (2) the amount of data that the query needs to process. For the analytical model of a single query, the response time of the query can be obtained by clarifying the above two points. For the analytical model of parallel query, additional research is needed (3) the contention pattern of resources between parallel queries, and the mode is explicitly described. The research process of the analytical model is also the research progress of the above three core contents. In 2004, Chaudhuri et al. [16] and Luo et al. [24] published a paper on query progress indicators at SIGMOD 2004, and studied the progress indicator and the query response time prediction model based on the analysis method.

The workload prediction has to do with query interaction in a mix that is known as building query progress indicator, statistical prediction models and analytical prediction models. Progress indicator is a tool which shows the percentage progress of a running query [15, 16]. The progress indicators fundamentally partition a query plan do on different phases and updated the query progress information based on the execution information consistently.

Ahmad et al. [10] study the problem for estimating simultaneously query response time. In [13], the authors propose an experiment driven approach for sampling. In the paper the authors use Gaussian process as the particular statistical model. The key restraint of the work is assumed static workload that is not practically. According to our reading, we indicate the concurrent query response time estimation problem under dynamic workload.

In database system the research communities have gained substantial interest to predicting query execution time [11,12, 13,14]. The current query response time prediction model is divided into analytical type and statistical type. Analytical modeling predicts response time by studying the query execution process. Statistical modeling uses machine learning methods to model query response time. This method can strike a balance between model complexity and usability. The static modeling technique is employed to estimate running time for a query in a database system [17,18,19]. We employed our developed Gscheduler by solving a linear programming problem and also for the estimation model to work [1].

III. THE FRAMEWORK

Estimation framework is described in this Section. In first instance, estimation problem is described, then solution is described and would be provided some analysis.

We present problem definition in Subsection A. Subsection B presents the structure of our defined workflow. Subsection C presents performance static model. Subsection C presents dynamic model.

A. The Problem Definition

In order to meet and maintained the peak performance of our proposed scheduler in our prior work, that decrease total completion time of a business intelligent (BI) workload [1]. Now to solve another workload management problem that estimate the total running time of a workload. The database administrators want to knowing how much the workload will expect to run.

The database consistently runs query mixes according to multiprogramming level M queries draw as $W = \{< q_i, w_i > | i = 1,2,\dots,N\}$, w_i represents the number of queries q_i. Whenever a query is finished execution, the database system selects the query from workload W to form a new mix, and estimate leftover running time of newly formed mix. We require to find an approach which give us estimate running time of newly form mixes.

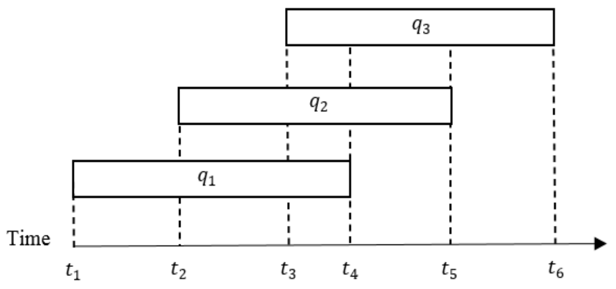

Fig 1. Illustration the Estimation Running Time Problem.

An example is employed to explain the estimation problem in Fig. 1, three queries q_1 , q_2 and q_3 that are concurrently running and arrive at time denoted as t_1 , t_2 and t_3. In this scenario, there are three estimation problems. At t_1 we need to estimate the running time of q_1. The estimation requires the information of the upcoming q_2, q_3 which is not available at t_1. At t_2 , q_2 join and need to make an estimation for both q_1 and q_2. Actually q_1 that has been running for some time, we require its remaining running time. The estimation requires the knowledge that q_3 will arrive which is unavailable at t_2. The same argument can be further applied to the estimation for q_1 , q_2 and q_3 at t_3. For example, let M be a mix of n queries $M = \{q_1, q_2, \ldots\ldots, q_n\}$ which are concurrently running. Let me know, s_0 is a start time of n queries and e_i is the end time. $T_i = e_i - s_0$ to be the execution time of q_i is defined. An estimation model is required to concern this problem. The estimation problem in Fig. 1 is generated by setting $M = \{q_1, q_2, q_3\}$ and $s_0 = t_3$.

B. The Structure of the Framework

The structure of the framework is initiated for batch workload. The structure contains a dynamic model that can estimate response time of the mixes. This nontrivial task is accomplished with dynamic model whosoever can estimate the running time of queries to manipulating performance static model.

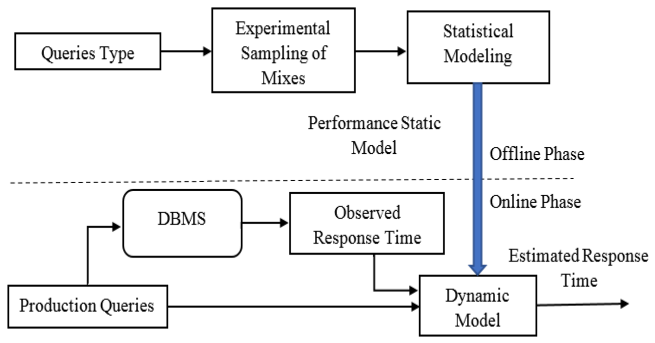

Fig 2. An Overview of the Framework.

Fig. 2 defines the overall workflow of the framework which consists of two major components, that are offline phase, and online phase. The workflow is used by a database administrator who executes the batch workloads repeatedly and need to estimate the total completion time of a workload. The database administrator provides a queries type q_i as input to the offline phase. Next, the detail description is described of the performance static model and the dynamic model.

C. The Performance Static Model

The impact of query interactions have been defined capturing by experimentally measuring the average completion time of various queries type. A new approach is proposed which designed and conducted experiments for possible query mix sampling. The experiment is run to chose query mix sample. Collecting the data from all experiments as a query mix model. No prior assumptions are required for this method about the working of the database system. For example, the database consistently executes queries concurrently drawn from the set of N queries in a mix, $m = \{q_1, q_2, \ldots\ldots, q_n\}$. Every executed query mix in the sample called a known mix. The response time of each query is recorded as the sample data set. A model is developed to estimate the newly formed unknown query mixes.

Through experiment driven approach, sampling experiments are required to collect data for static model. The sampling policy gives feasible query mixes that provides a feasible point for an offline phase. The method perks that the instances can be updated incessantly and improved model performance. Our workload generator generates query mixes sample. For collecting the samples, the workload setting is generated by client coordinator, the MPL=3. 19 hours is taken to run these experiments for 504 different query mixes of workload.

For the samples data, to develop a sampling algorithm is required, and then an appropriate regression model is employed, which gives accuracy in estimating query completion time in a mix. Now, a sampling algorithm is described for the efficiency of the sample data.

There are many types of sampling techniques. To choose the possible sampling from the 504 set of query mixes. The random sampling is employed, but according our modeling perspective, it is inefficient. The drawback of the random sampling is wide-ranging when to require to learn a good model. When we increase the queries type, the same sample data may be repeated unnecessarily that's why the sample data may be larged. For save the processing time, minimize the samples data is required for a large number of queries type.

For this purpose, Latine Hypercube Sampling (LHS) is used. McKay et al. [20] propose LHS that is suitable for designing of computer experiments. Stein et al. [21] prove that LHS is a powerful and useful method. LHS gives better coverage than random sampling. LHS comes from the family of space-filling designs and performs well in practice [22].

The constraints are harded that the number of concurrent query instances in a sampled mix would not be exceeded from the fixed multiprogramming level. The requirements can't be fulfilled by the simple LHS technique. For making it interactions aware and fixed conditions on MPL, an algorithm is needed to develop for collecting those sample data which have minimum estimation error and also to satisfy fixed instances of query types as fixed MPL.

The Task: A set of samples n mixes for the given query types T, MPL= M, interaction level = k, and permutation matrix = P_k.

The method:

- Let's P_k of size $n \times T$ by LHS, P_k is a query mix.
- P_k at random set values of $T - k$, the column set as 0 to k make the level of its interactions. The column does not exceed as fixed M.
- We want to cover the interactions level of k within n samples data. The n mixes that are allowed for k. We pick mixes from P_k.

The algorithm fulfills our requirements to sample mixes, which covers interactions level and makes the fixed Multiprogramming level. When a query finish execution and exits the new query comes and forms new mix constantly in the database, therefore a model is required which uses this performance static model and estimates the time of the newly formed mix. Next, we present detail of the model namely called dynamic model for estimating query response times.

D. The Dynamic Model

The dynamic model is very powerful for database system administration. The important feature of the dynamic model is a plugable incorporating of the scheduling method. The dynamic model is used for typical scheduling methods First-Come, First-Served, Shortest-Job-Next scheduling, and a variety of sophisticated scheduling methods.

Dynamic estimation problem: The workload, $W = \{< q_i, w_i > | i = 1,2,3, \ldots, N\}$, is run according to multiprogramming level M queries. w_i represents the number of queries q_i in a mix. Whenever a query is finished and exits, the database system selects the query from workload W to form a new mix, the model is estimated the remaining execution time of the mix according to the new mix.

For example, the query q_1 in Fig. 3, when m_1 start at time t_0. The model will estimate the leftover time of q_1 according to the query mix (q_1, q_3, q_6), from t_0 to t_4. When q_3 exits and q_4 comes in to form new mix m_2 , the model estimates the remaining execution time of q_1 according to mix (q_1, q_6, q_4), this is the time from t_1 to t_4. The estimation process for all other queries is the same.

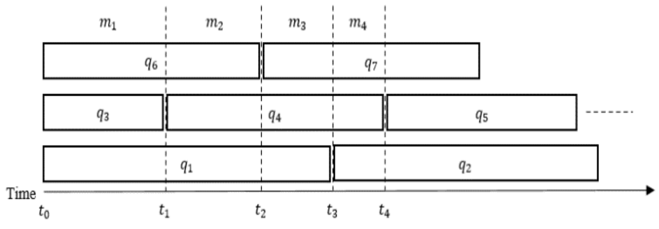

Fig 3. Different Mixes of Three Queries Running Continuously.

The dynamic model is similar to the progress indictors [5, 19, 26], but the dynamic model uses different modeling approach. (i) The dynamic model is activated at the exits of a query, but progress indicators predict periodically. (ii) The progress indicators are used in a single query scenario. The existing parallel progress indicators [5] is a simple extension of the classic progress indicators [17, 24]. The interactions between queries are not considered for the parallel progress indicators.

When a query exits, the dynamic model estimates the remaining execution time of remaining running queries using Equation as follows:

$$\hat{t}^{r}_{<q,m_s>} = t^{I} < q, m_s > \left(1 - tanh(\textstyle\sum_{l=1}^{s-1} \frac{t^{e}_{<q,m_l>}}{t^{I}_{<q,m_l>}})\right) \qquad (1)$$

$\hat{t}^{r}_{<q,m_s>}$, is shows the estimated remaining time of q with mix m_s. $t^I < q, m_s >$, is shows the running time of query q running in m_l. $t^e_{<q,m_l>}$, is the elapse time of query q in m_l. Equation 1 estimates the time needs to exit query q running in m_s . m_s estimates the starting time of the query instance. According to Fig. 3, suppose we need to estimate the remaining time of q_1 at t_2. The elapsed time of q_1 is t_0 to t_2, and the work done for q_1 is $\tanh\left(\frac{t^{e}_{\langle 1,\{6,3.1\rangle}}{t^{I}_{\langle 1,\{6,3.1\rangle}} + \frac{t^{e}_{\langle 1,\{6,4.1\rangle}}{t^{I}_{\langle 1,\{6,4.1\rangle}}\right)$. To estimate the remaining time of q_1 in mix (7, 4, 1), we multiply $t^{I}_{\langle 1,\{7,4.1\rangle}$ with $1 - tanh\left(\frac{t^{e}_{\langle 1,\{6,3.1\rangle}}{t^{I}_{\langle 1,\{6,3.1\rangle}} + \frac{t^{e}_{\langle 1,\{6,4.1\rangle}}{t^{I}_{\langle 1,\{6,4.1\rangle}}\right)$.

The dynamic model refreshes the state of unfinished queries at the end of the mix. When scheduling picks new query for making new mix, the dynamic model change the time state with the help of perfromance static model. That procedure continues running til all queries in the workload are completed.

$$L_w = \sum_{i=1}^{|W|-M+1} I_i$$

The workload estimated completion time is add the lengths for all mixes come upon throughout the imitation.

IV. Experiment Evaluation

This section presented experiment evaluation of the proposed approach. The estimation accuracy we measure in terms of mean relative error. Which is defined as:

$$\frac{1}{N}\sum_{i=1}^{N} \frac{|T_i^{est} - T_i^{act}|}{T_i^{act}}$$

The number of testing queries is *N*. The estimated and the actual execution time for q_i are T_i^{est} and T_i^{act}. We measured the estimation approaches as well. In subsection 5.7.1 gives out the experimental settings. In subsection 5.7.2 define the overall accuracy. In subsection 5.7.3 define the scheduling performance.

A. Experimental Setup

Environments. The software and hardware is described for using in our experiments. A machine with Intel E3500 2.7GHz CPU and 4GB RAM is used for experiment. PostgreSQL database is run under Window 2007x64. whole configurations of Postgres are the default and turned off entire the statistics and tuning tools. TPC-H scale factor 10, denoted by 10GB is used. The configuration advisor of the PostgreSQL ensures the configuration parameters are well acquainted.

Workload: Table III shown the workload of eighty instances of nine types of queries, which are chosen pursuant to the *TPC-H* and run on 10 GB database system. We limit the workload size and MPL by virtue of the long-term of queries in database system. Recall from Fig. 4 that 80 query workloads can take more than 6 hours to complete.

Methodology: We generated 80 workloads due to choice of queries type, use our Gscheduler algorithm [1] with other two scheduling algorithms as FCFS, SJF. The completion times limit from 6 hours to more than 8 hours. We comparison acutal time as *act* and estimated completion time as *est*. the estimation error is computed as.

$$estimation\ error = \frac{|est - act|}{act} \times 100$$

B. Overall Accuracy

The overall accuracy of our estimations. The relative estimation error in all 80 workloads in our experiment. In one case we use performance static model for *First-Come-First-Served*, second, we use *Shortest-Job-Next* and third we use *Gschedule*r. Amjad et.al. defined Gscheduler is to schedule query mixes for a given query workload in order to minimize *W*'s total completion time. When query finished in query mix the Gscheduler chose query from the workload which will take minimum time for execution. The estimation errors in all cases are less than 20% of the 80% time. Aginst the 80 workload queries, these results show that our framework is accurate for estimating total completion time of the workload.

The database administrator can accurate estimations for future workloads in a database with the help of one time samples.

TABLE III. WORKLOAD OF QUERIES

SF	Q_1	Q_4	Q_5	Q_6	Q_8	Q_{10}	Q_{12}	Q_{14}	Q_{18}
10	3	12	6	13	8	15	6	13	4

C. Scheduling Performance

Here, our Gscheduler is compared with two other state of the art scheduling approaches from the literature. The scheduling approaches are compared without any assistance from queueing models.

The performance of scheduling algorithms in this section and compare with each other, *First-Come-First-Served*, *Shortest-Job-Next* and *Gscheduler* scheduling algorithms. We represent the number of similar mixes used for the performance static model to estimate. The workload *W* contains 80 queries, as shown in Table III.

First-Come-First-Served is sensitive to the arrival order of queries in *W*, so we sequentially generate a sequence of queries and report the completion time of *W*. *Shortest-Job-Next* is not sensitive to the arrival order we select the shorted query based on query response time t_q in isolation. *Gscheduler* is different from *First -Come-First-Served*, *Shortest-Job-Next* scheduler, we randomly select a query from *W* whenever a query finish. The performance of all schedulers shows in Fig. 4, 5, 6 and 7, respectively.

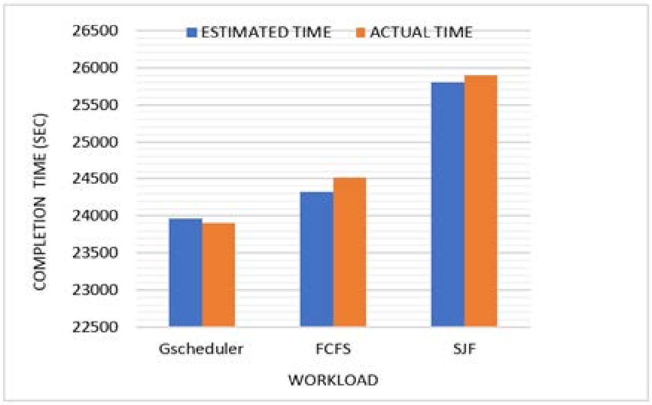

Fig 4. Completion Times for Gscheduler, FCFS and SJN.

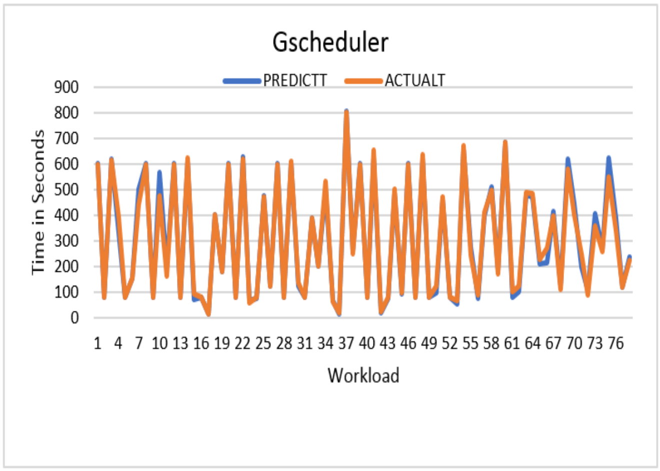

Fig 5. Predicted and Actual Time of the Gscheduler.

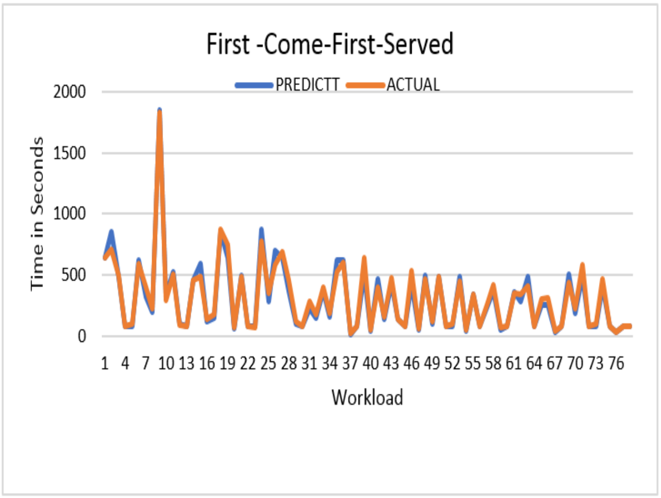

Fig 6. Predicted and Actual Time of the FCFS.

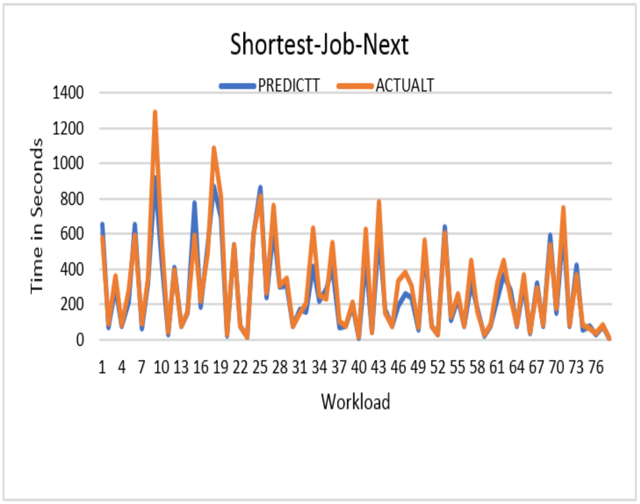

Fig 7. Predicted and Actual Time of the SJN.

V. Conclusion and Future Work

In a business intelligence setting, it is substantial for database administrators to estimate the total completion time of the workload. The state of the art methods don't tender any procedure for the database administrators to estimate the completion time of the workload. An approach, interaction into account is presented in this paper. Interaction aware experiment driven model assumbled with workload simulation for estimating completion time of the workload. an experimental evaluation with TPC-H benchmark running on PostgreSQL has proven that our approach can estimate workload completion time with high degree of accuracy across a broad spectrum of workloads.

A full research agenda is moving forward. We want to conduct research to form an integrated characterizing framework for query mixes, which may help us learn more about to measure query interactions method so that we could improve the models. We would make plan to study the adaptable similarity model which could make more accurate prediction because the DBMS is changing over time.

References

[1] Amjad, M. and J. Zhang, Gscheduler: A Query Scheduler Based on Query Interactions. In proceeding of the Web Information Systems and Applications, Lecture Notes in Computer Science, Springer, Cham, China: WISA, 2018: 11242.

[2] Wasserman, T, J. Martin, P. Skillicorn, D, B. and H. Rizvi, Developing a characterization of business intelligence workloads for sizing new database systems. In DOLAP, 2004.

[3] Mishra, C. and N. Koudas, The design of a query monitoring system. ACM Trans. Database Syst., 34(1), 2009.

[4] Guirguis, S. Sharaf, M. A. and P. K. Chrysanthis, A. Labrinidis, and K. Pruhs. Adaptive scheduling of web transactions. In ICDE, 2009.

[5] Tozer, S. Brecht, T. and A. Aboulnaga, Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads[C]. Proc. of the 26th International Conference on Data Engineering. Long Beach, California, USA: IEEE, 2010:397-408.

[6] Akdere, M. Çetintemel, U. Riondato, M. et al., Learning-based Query Performance Modeling and Prediction [C]// Proc. of the 25th International Conference on Data Engineering. Washington, DC, USA: IEEE, 2012: 390-401.

[7] Ganapathi, A. Kuno, H. Dayal, U. and J. L. Wiener, et al., Predicting multiple metrics for queries: Better decisions enabled by machine learning[C] // Proc. of the Int Conf Data Eng, 2009: 592–603.

[8] Li, J. K ¨ onig, A. C. Narasayya, V. R and S. Chaudhuri, Robust estimation of resource consumption for sql queries using statistical techniques. PVLDB, 5(11):1555–1566, 2012.

[9] Wu, W. Yun, C. Shenghuo, Z. and T. Jun'ichi, et al., Predicting Query Execution Time: Are Optimizer Cost Models Really Unusable? Proc. of the 29th International Conference on Data Engineering, Brisbane Computer Society, Australia: IEEE, 2013: 1081-1092.

[10] M. Ahmad, M. Duan, S. Aboulnaga, A. and S. Babu, Predicting completion times of batch query workloads using interaction-aware models and simulation. In EDBT, pages 449–460, 2011.

[11] Duggan, J. Cetintemel, U. Papaemmanouil, O. et al., Performance prediction for concurrent database workloads. In Proceeding of the ACM SIGMOD International Conference on Management of Data. Athens, Greece: ACM, 2011:337-348.

[12] Ahmad, M. Aboulnaga, A. Babu, S. and K. Munagala, Interaction-aware scheduling of report-generation workloads. The VLDB Journal, 20:589–615, 2011.

[13] Ahmad, M. Duan, S. Aboulnaga, A and S. Babu, Predicting completion times of batch query workloads using interaction-aware models and simulation. In EDBT, pages 449–460, 2011.

[14] Wu, W. Chi, Y. Zhu, S. Tatemura, J. Hacıg¨ um ¨ us, H. and J. F. Naughton, Predicting query execution time: are optimizer cost models really unusable? In ICDE, 2013.

[15] Li, J. Nehme, R.V. and JF. Naughton, GSLPI: a Cost-based Query Progress Indicator. Proc. of the 28th International Conference on Data Engineering, 2012, Washington DC, USA, IEEE Computer Society, pp. 678-689.

[16] Chaudhuri, S. Narasayya, V. R. and R. Ramamurthy, Estimating Progress of Execution for SQL Queries. Proc. of the 2004 ACM SIGMOD/PODS Conference ,2004, Paris, France, ACM, pp. 803-814.

[17] Gupta, C. Mehta, A. and U. Dayal, PQR: Predicting query execution times for autonomous workload management. In Proceedings of the 5th International Conference on Autonomic Computing (ICAC), 2008.

[18] Babu, S. Borisov, N. Duan, S. Herodotou, H and V. Thummala, Automated experiment-driven management of (database) systems. In Proceedings of the 12th Workshop on Hot Topics in Operating Systems (HotOS), 2009.

[19] Suri, R. Sahu, S. and M. Vernon, Approximate mean value analysis for closed queuing networks with multiple-server stations. In proceeding. Of Industrial Engineering Research Conference, 2007.

[20] McKay, M. D. Conover, W. J. and R. J. Beckman, A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code, Technometrics, 1979, 21,239-245.

[21] M. Stein, Large Sample Properties of Simulations Using Latin Hypercube Sampling, Technometrics, 1987 29:2, 143-151, DOI: 10.1080/00401706.1987.10488205.

[22] Hicks, C. R. and K. V. Turner, Fundamental Concepts in the Design of Experiments. Oxford University Press, 1999.

[23] Ahmad, M., Aboulnaga, A., Babu, S., and Munagala, K. Modeling and Exploiting Query Interactions in Database Systems . Proc. of the 17th ACM Conference on Information and Knowledge Management, 2008, Napa Valley, California, USA, ACM, pp. 183-192.

[24] Luo, G. Naughton, J. F. Ellmann, C. J, et al. Toward a progress indicator for database queries[C] // Proc. of the ACM SIGMOD International Conference on Management of Data. Paris, France: ACM, 2004:791-802.